PARTICLE SYSTEMS

AIM OF THE PROJECT:

This project implements particle systems using OpenGL. The various particle systems modeled are fire, smoke, snow, rain, water, lava, explosion and fountains. Apart from this, height maps are used to render a textured volcano.

INTRODUCTION- The basics

A particle models some real world entity. It has various attributes depending on the entity it models. The attributes of the particle used in this program are

1. age
2. Initial Velocity
3. current velocity
4. initial displacement
5. position
6. initial color
7. final color
8. transparency
9. acceleration
10. lifetime
11. initial size
12. final size

A particle system models a collection of particles and manages their dynamics. The most important attribute of a particle system is the number of particles. A particle system keeps adding and removing particles and changing their value in different ways to model different real world systems. Normally, The age of the particle is incremented each time it is rendered and when the age is greater than its designated lifetime, the particle is removed from the system.

A particle engine is a program that creates and destroys particle systems based on what is required in the program.
MODELLING VARIOUS PARTICLE SYSTEMS:

An important factor that makes particle systems look real is randomness.

*Randomness in velocity:* This is incorporated by choosing $\Theta$ and $\Phi$ (the normal spherical coordinates) randomly from a given range and calculating the components of velocity from them.

Randomness is introduced in position, color, lifespan too by choosing a random number from a particular range.

**Fire:**

The key feature of this particle system is the fact that the varying temperature of the flame is modeled by blending individual particles. At the core of the fire, there are more particles. So when more particles are blended, we get a brighter shade. Near the periphery, there are fewer particles. So we get a dull shade.

Apart from this each particle is assigned a start color and a final color (which it has when it dies). In each interval, the color of the particle is varied to make the fire more realistic. There is a gradual change of color from the start color to the final color.

Also, The initial velocity of each particle in the fire is constrained to deviate not more than 30 degrees from the y axis.
Fountains

Here each particle is given some downward acceleration to model gravity effect. So particles follow a parabolic path and hit the ground.

When a droplet hits the ground, ideally, we would have to model the way it flows on the ground. But since this is difficult, a puddle of water has been created at the base of the fountain as a compromise.

The pictures at the bottom were created by two fountain particle systems with different acceleration values.
Smoke:

The initial value of each particle is chosen to be a very light shade of gray and the final color is chosen to be the background color (in this case blue). A picture of smoke is shown below.
**Rain:**

The initial velocity in this case is constrained to be in a downward direction. Some downward acceleration is given to make the scene more realistic.

An important thing that must normally be handled in the case of rain the flow of water when it hits the ground. Due to constraints of time this has been omitted. The figure below shows rain on a textured ground plane.

![Rain on a textured ground plane](image)

**Snow:**

Here, the particles are not given any velocity and are given very small or no acceleration. An important thing to be handled here is the way snow is accumulated when it hits the ground. This has been attempted but has not been fully handled as multiple snow particles cannot be accumulated in this program.
Accumulation of Snow:

Another simple hack to make snow more appealing would be to just color the terrain white. In fact, a simple hack to model accumulation of snow would be to just generate height map for the scene to give the illusion of lumps of snow at certain places.
Explosions:

An explosion can be modeled by choosing an initial velocity in a totally random way. The following figure shows explosion modeled in the background of a mountain rendered using a height map.

Lava:

This is modeled in a way similar to fire. The effect of the figure below is supposed to be that of a volcano.
USING HEIGHTMAPS FOR TERRAIN RENDERING

A height map is just the listing of height values for each point in the picture. The program uses a height map by reading the raw data from a file, storing it in an array and reading the appropriate height value from it.

The following figure shows a terrain rendered using a height map and a texture mapped onto it.
IMPLEMENTATION NOTE:

An individual particle is implemented as a class with all the above mentioned attributes and certain methods to manage its state.

Due to the similarity in the implementation of various particle systems, a base class particle system can be created. This has various subclasses which have methods overriding the base class. (For instance the particle system for snow may render in a slightly different way as it must take care of the accumulation).

USER INTERFACE:
The user interface provided has the following
1. Checkboxes to select particle systems.
   Click on a check box to choose one particular type of particle system and click on start/stop to actually create the system. Click again on the system and click on start/stop to destroy the system.
2. Translation controls to change the eye point and edit boxes to set the parameters of GluLookAt
3. A set if attributes of the particle system that can be changed.
   When we click on a check box for a particular particle system, the corresponding particle system is activated and its attributes are shown. They can be edited and then clicking on the start/stop button starts that particle system with attributes we set.

POSSIBLE EXTENSIONS:

1. Modelling the accumulation of snow more accurately.
2. Use this approach to generate other particle systems like trees, forests etc…

REFERENCES:

3. W. T. Reeves, "Particle Systems - A Technique for Modeling a Class of Fuzzy Objects".